

# Transductive Pattern Learning for Information Extraction

**Brian McLernon    Nicholas Kushmerick**

School of Computer Science and Informatics  
University College Dublin, Ireland  
{brian.mclernon,nick}@ucd.ie

## Abstract

The requirement for large labelled training corpora is widely recognized as a key bottleneck in the use of learning algorithms for information extraction. We present TPLEX, a semi-supervised learning algorithm for information extraction that can acquire extraction patterns from a small amount of labelled text in conjunction with a large amount of unlabelled text. Compared to previous work, TPLEX has two novel features. First, the algorithm does not require redundancy in the fragments to be extracted, but only redundancy of the extraction patterns themselves. Second, most bootstrapping methods identify the highest quality fragments in the unlabelled data and then assume that they are as reliable as manually labelled data in subsequent iterations. In contrast, TPLEX's scoring mechanism prevents errors from snowballing by recording the reliability of fragments extracted from unlabelled data. Our experiments with several benchmarks demonstrate that TPLEX is usually competitive with various fully-supervised algorithms when very little labelled training data is available.

## 1 Introduction

Information extraction is a form of shallow text analysis that involves identifying domain-specific fragments within natural language text. Most recent research has focused on learning algorithms that automatically acquire extraction patterns from manually labelled training data (e.g., (Riloff, 1993; Califf and Mooney, 1999; Soderland, 1999; Freitag and Kushmerick, 2000; Ciravegna, 2001; Finn and Kushmerick, 2004)). This training data takes the form of the original text, annotated with the fragments to be extracted.

Due to the expense and tedious nature of this labelling process, it is widely recognized that a key bottleneck in deploying such algorithms is the need to create a sufficiently large training corpus for each new domain. In response to this challenge, many researchers have investigated semi-supervised learning algorithms that learn from a (relatively small) set of labelled texts

in conjunction with a (relatively large) set of unlabelled texts (e.g., (Riloff, 1996; Brin, 1998; Yangarber et al., 2002)).

In this paper, we present TPLEX, a semi-supervised algorithm for learning information extraction patterns. The key idea is to exploit the following recursive definitions: good patterns extract good fragments, and good fragments are extracted by good patterns. To operationalize this recursive definition, we initialize the pattern and fragment scores with labelled data, and then iterate until the scores have converged.

Most prior semi-supervised approaches to information extraction assume that fragments are essentially named entities, so that there will be many occurrences of any given fragment. For example, for the task of discovering diseases ("influenza", "Ebola", etc), prior algorithms assume that each disease will be mentioned many times, and that every occurrence of such a disease in unlabelled text should be extracted. However, it may not be the case that fragments to be extracted occur more than once in the corpus, or that every occurrence of a labelled fragment should be extracted. For example, in the well-known CMU Seminars corpus, any given person usually gives just one seminar, and a fragment such as "3pm" sometimes indicates the start time, other occurrences indicate the end time, and some occurrence should not be extracted at all. Rather than relying on redundancy of the fragments, TPLEX exploits redundancy of the learned extraction patterns.

TPLEX is a transductive algorithm (Vapnik, 1998), in that the goal is to perform extraction from a given unlabelled corpus, given a labelled corpus. This is in contrast to the typical machine learning framework, where the goal is a set of extraction patterns (which can of course then be applied to new unlabelled text). As a side-effect, TPLEX does generate a set of extraction patterns which may be a useful in their own right, depending on the application.

We have compared TPLEX with various competitors on a variety of real-world extraction tasks. We have observed that TPLEX's performance matches or exceeds these in several benchmark tasks. The remainder of this paper is organized as follows. After describing related work in more detail (Sec. 2), we describe the TPLEX algorithm (Sec. 3). We then discuss a series of experiments to compare TPLEX with various supervised al-

Report Documentation Page			Form Approved OMB No. 0704-0188		
Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.					
1. REPORT DATE <b>2006</b>		2. REPORT TYPE		3. DATES COVERED <b>00-00-2006 to 00-00-2006</b>	
4. TITLE AND SUBTITLE <b>Transductive Pattern Learning for Information Extraction</b>			5a. CONTRACT NUMBER		
			5b. GRANT NUMBER		
			5c. PROGRAM ELEMENT NUMBER		
6. AUTHOR(S)			5d. PROJECT NUMBER		
			5e. TASK NUMBER		
			5f. WORK UNIT NUMBER		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) <b>School of Computer Science and Informatics,University College Dublin,Dublin, Ireland, ,</b>			8. PERFORMING ORGANIZATION REPORT NUMBER		
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)			10. SPONSOR/MONITOR'S ACRONYM(S)		
			11. SPONSOR/MONITOR'S REPORT NUMBER(S)		
12. DISTRIBUTION/AVAILABILITY STATEMENT <b>Approved for public release; distribution unlimited</b>					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT					
15. SUBJECT TERMS					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES <b>7</b>	19a. NAME OF RESPONSIBLE PERSON
a. REPORT <b>unclassified</b>	b. ABSTRACT <b>unclassified</b>	c. THIS PAGE <b>unclassified</b>			

gorithms (Sec. 4). We conclude with a summary of observations made during the evaluation, and a discussion of future work (Sec. 5).

## 2 Related work

A review of machine learning for information extraction is beyond the scope of this paper; see e.g. (Cardie, 1997; Kushmerick and Thomas, 2003).

A number of researchers have previously developed bootstrapping or semi-supervised approaches to information extraction, named entity recognition, and related tasks (Riloff, 1996; Brin, 1998; Riloff and Jones, 1999; Agichtein et al., 2001; Yangarber et al., 2002; Stevenson and Greenwood, 2005; Etzioni et al., 2005).

Several approaches for learning from both labeled and unlabeled data have been proposed (Yarowsky, 1995; Blum and Mitchell, 1998; Collins and Singer, 1999) where the unlabeled data is utilised to boost the performance of the algorithm. In (Collins and Singer, 1999) Collins and Singer show that unlabeled data can be used to reduce the level of supervision required for named entity classification. However, their approach is reliant on the presence of redundancy in the named entities to be identified.

TPLEX is most closely related to the NOMEN algorithm (Yangarber et al., 2002). NOMEN has a very simple iterative structure: at each step, a very small number of high-quality new fragments are extracted, which are treated in the next step as equivalent to seeds from the labeled documents. NOMEN has a number of parameters which must be carefully tuned to ensure that it does not over-generalise. Erroneous additions to the set of trusted fragments can lead to a snowballing of errors.

Also, NOMEN uses a binary scoring mechanism, which works well in dense corpora with substantial redundancy. However, many information extraction tasks feature sparse corpora with little or no redundancy. We have extended NOMEN by allowing it to make finer-grained (as opposed to binary) scoring decisions at each iteration. Instead of definitively assigning a position to a given field, we calculate the likelihood that it belongs to the field over multiple iterations.

## 3 The TPLEX algorithm

The goal of the algorithm is to identify the members of the target fields within unlabelled texts by generalizing from seed examples in labelled training texts. We achieve this by generalizing boundary detecting patterns and scoring them with a recursive scoring function.

As shown in Fig. 1, TPLEX bootstraps the learning process from a seed set of labelled examples. The examples are used to populate initial pattern sets for each target field, with patterns that match the start and end positions of the seed fragments. Each pattern is then generalised to produce more patterns, which are in turn

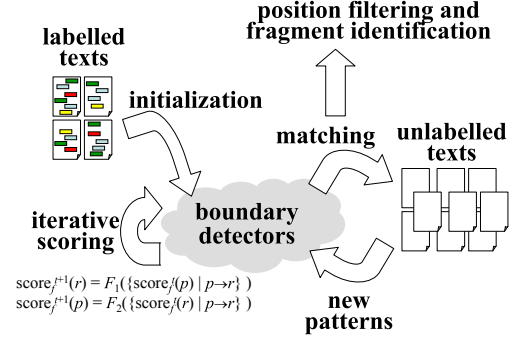


Figure 1: An overview of TPLEX. A key idea of the algorithm is the following recursive scoring method: pattern scores are a function of the scores of the positions they extract, and position scores are a function of the scores of the patterns that extract them.

applied to the corpus in order to identify more base patterns. This process iterates until no more patterns can be learned.

TPLEX employs a recursive scoring metric in which good patterns reinforce good positions, and good positions reinforce good patterns. Specifically, we calculate confidence scores for positions and patterns. Our scoring mechanism calculates the score of a pattern as a function of the scores of the positions that it matches, and the score of a position as a function of the scores of the patterns that extract it.

TPLEX is a multi-field extraction algorithm in that it extracts multiple fields simultaneously. By doing this, information learned for one field can be used to constrain patterns learned for others. Specifically, our scoring mechanism ensures that if a pattern scores highly for one field, its score for all other fields is reduced.

In the remainder of this section, we describe the algorithm by formalizing the space of learned patterns, and then describing TPLEX’s scoring mechanism.

### 3.1 Boundary detection patterns

TPLEX extracts fragments of text by identifying probable fragment start and end positions, which are then assembled into complete fragments. TPLEX’s patterns are therefore *boundary detectors* which identify one end of a fragment or the other. TPLEX learns patterns to identify the start and end of target occurrences independently of each other. This strategy has previously been employed successfully (Freitag and Kushmerick, 2000; Ciravegna, 2001; Yangarber et al., 2002; Finn and Kushmerick, 2004).

TPLEX’s boundary detectors are similar to those learned by BWI (Freitag and Kushmerick, 2000). A boundary detector has two parts, a left pattern and a right pattern. Each of these patterns is a sequence of tokens, where each is either a literal or a generalized token. For example, the boundary detector

[will be <punc>][<caps> <fname>]  
 would correctly find the start of a name in an utterance such as “will be: Dr Robert Boyle” and “will be, Sandra Frederick”, but it will fail to identify the start of the name in “will be Dr. Robert Boyle”. The boundary detectors that find the beginnings of fragments are called the pre-patterns, and the detectors that find the ends of fragments are called the post-patterns.

### 3.2 Pattern generation

As input, TPLEX requires a set of tagged seed documents for training, and an untagged corpus for learning. The seed documents are used to initialize the pre-pattern and post-pattern sets for each of the target fields. Within the seed documents each occurrence of a fragment belonging to any of the target categories is surrounded by a set of special tags that denote the field to which it belongs.

The algorithm parses the seed documents and identifies the tagged fragments in each document. It then generates patterns for the start and end positions of each fragment based on the surrounding tokens.

Each pattern varies in length from 2 to  $n$  tokens. For a given pattern length  $\ell$ , the patterns can then overlap the position by zero to  $\ell$  tokens. For example, a pre-pattern of length four with an overlap of one will match the three tokens immediately preceding the start position of a fragment, and the one token immediately following that position. In this way, we generate  $\sum_{i=2}^n (i+1)$  patterns from each seed position. In our experiments, we set the maximum pattern length to be  $n = 4$ .

TPLEX then grows these initial sets for each field by generalizing the initial patterns generated for each seed position. We employ eight different generalization tokens when generalizing the literal tokens of the initial patterns. The wildcard token `<*>` matches every literal. The second type of generalization is `<punc>`, which matches punctuation such as commas and periods. Similarly, the token `<caps>` matches literals with an initial capital letter, `<num>` matches a sequence of digits, `<alpha_num>` matches a literal consisting of letters followed by digits, and `<num_alpha>` matches a literal consisting of digits followed by letters. The final two generalisations are `<fname>` and `<lname>`, which match literals that appear in a list of first and last names (respectively) taken from US Census data.

All patterns are then applied to the entire corpus, including the seed documents. When a pattern matches a new position, the tokens at that position are converted into a maximally-specialized pattern, which is added to the pattern set. Patterns are repeatedly generalized until only one literal token remains. This whole process iterates until no new patterns are discovered. We do not generalize the new maximally-specialized patterns discovered in the unlabelled data. This ensures that all patterns are closely related to the seed data. (We experimented with generalizing patterns from the unlabelled

data, but this rapidly leads to overgeneralization.)

The locations in the corpus where the patterns match are regarded as potential target positions. Pre-patterns indicate potential start positions for target fragments while post-patterns indicate end positions. When all of the patterns have been matched against the corpus, each field will have a corresponding set of potential start and end positions.

### 3.3 Notation & problem statement

Positions are denoted by  $r$ , and patterns are denoted by  $p$ . Formally, a pattern is equivalent to the set of positions that it extracts. The notation  $p \rightarrow r$  indicates that pattern  $p$  matches position  $r$ . Fields are denoted by  $f$ , and  $F$  is the set of all fields.

The labelled training data consists of a set of positions  $R = \{\dots, r, \dots\}$ , and a labelling function  $T : R \rightarrow F \cup \{X\}$  for each such position.  $T(r) = f$  indicates that position  $r$  is labelled with field  $f$  in the training data.  $T(r) = X$  means that  $r$  is not labelled in the training data (i.e.  $r$  is a negative example for all fields).

The unlabelled test data consists of an additional set of positions  $U$ .

Given this notation, the learning task can be stated concisely as follows: extend the domain of  $T$  to  $U$ , i.e. generalize from  $T(r)$  for  $r \in R$ , to  $T(r)$  for  $r \in U$ .

### 3.4 Pattern and position scoring

When the patterns and positions for the fields have been identified we must score them. Below we will describe in detail the recursive manner in which we define  $\text{score}_f(r)$  in terms of  $\text{score}_f(p)$ , and vice versa. Given that definition, we want to find fixed-point values for  $\text{score}_f(p)$  and  $\text{score}_f(r)$ . To achieve this, we initialize the scores, and then iterate through the scoring process (i.e. calculate scores at step  $t+1$  from scores at step  $t$ ). This process repeats until convergence.

**Initialization.** As the scores of the patterns and positions of a field are recursively dependant, we must assign initial scores to one or the other. Initially the only elements that we can classify with certainty are the seed fragments. We initialise the scoring function by assigning scores to the positions for each of the fields. In this way it is then possible to score the patterns based on these initial scores.

From the labelled training data, we derive the prior probability  $\pi(f)$  that a randomly selected position belongs to field  $f \in F$ :

$$\pi(f) = |\{r \in R \mid T(r) = f\}|/|R|.$$

Note that  $1 - \sum_f \pi(f)$  is simply the prior probability that a randomly selected position should not be extracted at all; typically this value is close to 1.

Given the priors  $\pi(f)$ , we score each potential posi-

tion  $r$  in field  $f$ :

$$\text{score}_f^0(r) = \begin{cases} \pi(f) & \text{if } r \in U, \\ 1 & \text{if } r \in R \wedge T(r) = f, \text{ and} \\ 0 & \text{if } r \in R \wedge T(r) \neq f. \end{cases}$$

The first case handles positions in the unlabelled documents; at this point we don't know anything about them and so fall back to the prior probabilities. The second and third cases handle positions in the seed documents, for which we have complete information.

**Iteration.** After initializing the scores of the positions, we begin the iterative process of scoring the patterns and the positions. To compute the score of a pattern  $p$  for field  $f$  we compute a positive score,  $\text{pos}_f(p)$ ; a negative score,  $\text{neg}_f(p)$ ; and an unknown score,  $\text{unk}(p)$ .  $\text{pos}_f(p)$  can be thought of as a measure of the benefit of  $p$  to  $f$ , while  $\text{neg}_f(p)$  measures the harm of  $p$  to  $f$ , and  $\text{unk}(p)$  measures the uncertainty about the field with which  $p$  is associated.

These quantities are defined as follows:  $\text{pos}_f(p)$  is the average score for field  $f$  of positions extracted by  $p$ . We first compute:

$$\text{pos}_f(p) = \frac{1}{Z_p} \sum_{p \rightarrow r} \text{score}_f^t(r),$$

where  $Z_p = \sum_f \sum_{p \rightarrow r} \text{score}_f^t(r)$  is a normalizing constant to ensure that  $\sum_f \text{pos}_f(p) = 1$ .

For each field  $f$  and pattern  $p$ ,  $\text{neg}_f(p)$  is the extent to which  $p$  extracts positions whose field is not  $f$ :

$$\text{neg}_f(p) = 1 - \text{pos}_f(p).$$

Finally,  $\text{unk}(p)$  measures the degree to which  $p$  extract positions whose field is unknown:

$$\text{unk}(p) = \frac{1}{|\{p \rightarrow r\}|} \sum_{p \rightarrow r} \text{unk}(r),$$

where  $\text{unk}(r)$  measures the degree to which position  $r$  is unknown. To be completely ignorant of a position's field is to fall back on the prior field probabilities  $\pi(f)$ . Therefore, we calculate  $\text{unk}(r)$  by computing the sum of squared differences between  $\text{score}_f^t(r)$  and  $\pi(f)$ :

$$\begin{aligned} \text{unk}(r) &= 1 - \frac{1}{Z} \text{SSD}(r), \\ \text{SSD}(r) &= \sum_f (\text{score}_f^t(r) - \pi(f))^2, \\ Z &= \max_r \text{SSD}(r). \end{aligned}$$

The normalization constant  $Z$  ensures that  $\text{unk}(r) = 0$  for the position  $r$  whose scores are the most different from the priors—ie,  $r$  is the “least unknown” position.

For each field  $f$  and pattern  $p$ ,  $\text{score}_f^{t+1}(p)$  is defined in terms of  $\text{pos}_f(p)$ ,  $\text{neg}_f(p)$  and  $\text{unk}(p)$  as follows:

$$\begin{aligned} \text{score}_f^{t+1}(p) &= \frac{\text{pos}_f(p)}{\text{pos}_f(p) + \text{neg}_f(p) + \text{unk}(p)} \cdot \text{pos}_f(p) \\ &= \frac{\text{pos}_f(p)^2}{1 + \text{unk}(p)} \end{aligned}$$

This definition penalizes patterns that are either inaccurate or have low coverage.

Finally, we complete the iterative step by calculating a revised score for each position:

$$\text{score}_f^{t+1}(r) = \begin{cases} \text{score}_f^t(r) & \text{if } r \in R \\ \frac{\sum_{p \rightarrow r} \text{score}_f^t(p) - \min}{\max - \min} & \text{if } r \in U, \end{cases}$$

where  $\min = \min_{f, p \rightarrow r} \sum_{p \rightarrow r} \text{score}_f^t(p)$  and  $\max = \max_{f, p \rightarrow r} \sum_{p \rightarrow r} \text{score}_f^t(p)$ , are used to normalize the scores to ensure that the scores of unlabelled positions never exceed the scores of labelled positions. The first case in the function for  $\text{score}_f^{t+1}(r)$  handles positive and negative seeds (i.e. positions in labelled texts), the second case is for unlabelled positions.

We iterate this procedure until the scores of the patterns and positions converge. Specifically, we stop when

$$\sum_f \left( \frac{\sum_p |\text{score}_f^t(p) - \text{score}_f^{t-1}(p)|^2}{\sum_r |\text{score}_f^t(r) - \text{score}_f^{t-1}(r)|^2} \right) < \theta.$$

In our experiments, we fixed  $\theta = 1$ .

### 3.5 Position filtering & fragment identification

Due to the nature of the pattern generation strategy, many more candidate positions will be identified than there are targets in the corpus. Before we can proceed with matching start and end positions to form fragments, we filter the positions to remove the weaker candidates.

We rank all of positions for each field according to their score. We then select positions with a score above a threshold  $\beta$  as potential positions. In this way we reduce the number of candidate positions from tens of thousands to a few hundred.

The next step in the process is to identify complete fragments within the corpus by matching pre-positions with post-positions. To do this we compute the length probabilities for the fragments of field  $f$  based on the lengths of the seed fragments of  $f$ . Suppose that position  $r_1$  has been identified as a possible start for field  $f$ , and position  $r_2$  has been identified as a possible field  $f$  end, and let  $P_f(\ell)$  be the fraction of field  $f$  seed fragments with length  $\ell$ . Then the fragment  $e = (r_1, r_2)$  is assigned a score

$$\text{score}_f(e) = \text{score}_f(r_1) \cdot \text{score}_f(r_2) \cdot P_f(r_2 - r_1 + 1).$$

Despite these measures, overlapping fragments still occur. Since the correct fragments can not overlap, we know that if two extracted fragments overlap, at least one must be wrong. We resolve overlapping fragments by calculating the set of non-overlapping fragments that maximises the total score while also accounting for the expected rate of occurrence of fragments from each field in a given document.

In more detail, let  $E$  be the set of all fragments extracted from some particular document  $D$ . We are interested in the score of some subset  $G \subseteq E$  of  $D$ 's

fragments. Let  $\text{score}(G)$  be the chance that  $G$  is the correct set of fragments for  $D$ . Assuming that the correctness of the fragments can be determined independently *given* that the correct number of fragments have been identified for each field, then  $\text{score}(G)$  can be defined zero if  $\exists (r_1, r_1), (s_2, r_2) \in G$  such that  $(s_1, r_1)$  overlaps  $(s_2, r_2)$ , and  $\text{score}(G) = \prod_f \text{score}(G_f)$  otherwise, where  $G_f \subseteq G$  is the fragments in  $G$  for field  $f$ . The score of  $G_f = \{e^1, e^2, \dots\}$  is defined as  $\text{score}(G_f) = \Pr(|G_f|) \cdot \prod_j \text{score}(e^j)$ , where  $\Pr(|G_f|)$  is the fraction of training documents that have  $|G_f|$  instances of field  $f$ .

It is infeasible to enumerate all subsets  $G \subseteq E$ , so we perform a heuristic search. The states in the search space are pairs of the form  $(G, P)$ , where  $G$  is a list of *good* fragments (i.e. fragments that have been accepted), and  $P$  is a list of *pending* fragments (i.e. fragments that haven't yet been accepted or rejected).

The search starts in the state  $(\{\}, E)$ , and states of the form  $(G, \{\})$  are terminal states. The children of state  $(G, P)$  are all ways to move a single fragment from  $P$  to  $G$ . When forming a child's pending set, the moved fragment along with all fragments that it overlaps are removed (meaning that the moved fragment is selected and all fragments with which it overlaps are rejected). More precisely, the children of state  $(G, P)$  are:

$$\left\{ (G', P') \mid \begin{array}{l} e \in P \wedge G' = G \cup \{e\} \wedge \\ P' = \{e' \in P \mid e' \text{ doesn't overlap } e\} \end{array} \right\}.$$

The search proceeds as follows. We maintain a set  $S$  of the best  $K$  non-terminal states that are to be expanded, and the best terminal state  $B$  encountered so far. Initially,  $S$  contains just the initial state. Then, the children of each state in  $S$  are generated. If there are no such children, the search halts and  $B$  is returned as the best set of fragments. Otherwise,  $B$  is updated if appropriate, and  $S$  is set to the best  $K$  of the new children. Note that  $K = \infty$  corresponds to an exhaustive search. In our experiments, we used  $K = 5$ .

## 4 Experiments

To evaluate the performance of our algorithm we conducted experiments with four widely used corpora: the CMU seminar set, the Austin jobs set, the Reuters acquisition set [www.isi.edu/info-agents/RISE], and the MUC-7 named entity corpus [www ldc.upenn.edu].

We randomly partitioned each of the datasets into two evenly sized subsets. We then used these as labeled and unlabeled sets. In each experiment the algorithm was presented with a document set comprised of the test set and a randomly selected percentage of the documents from the training set. For example, if an experiment involved providing the algorithm with a 5% seed set, then 5% of the documents in the training set (2.5% of the documents in the entire dataset) would be selected at random and used in conjunction with the documents in the test set.

For each training set size, we ran five iterations with a randomly selected subset of the documents used for training. Since we are mainly motivated by scenarios with very little training data, we varied the size of the training set from 1–10% (1–16% for MUC-7NE) of the available documents. Precision, recall and F1 were calculated using the BWI (Freitag and Kushmerick, 2000) scorer. We used *all occurrences* mode, which records a match in the case where we extract all of the valid fragments in a given document, but we get no credit for partially correct extractions.

We compared TPLEX to BWI (Freitag and Kushmerick, 2000), LP<sup>2</sup> (Ciravegna, 2001), ELIE (Finn and Kushmerick, 2004), and an approach based on conditional random fields (Lafferty et al., 2001). The data for BWI was obtained using the TIES implementation [tcc.itc.it/research/textec/tools-resources/ties.html]. The data for the LP<sup>2</sup> learning curve was obtained from (Ciravegna, 2003). The results for ELIE were generated by the current implementation [http://smi.ucd.ie/aidan/Software.html]. For the CRF results, we used MALLET's SimpleTagger (McCallum, 2002), with each token encoded with a set of binary features (one for each observed literal, as well as the eight token generalizations).

Our results in Fig. 2 indicate that in Acquisitions dataset, our algorithm substantially outperforms the competitors at all points on the learning curve. For the other datasets, the results are mixed. For SA and Jobs, TPLEX is the second best algorithm at the low end of the learning curve, and steadily loses ground as more labelled data is available. TPLEX is the least accurate algorithm for the MUC data. In Sec. 5, we discuss a variety of modifications to the TPLEX algorithm that we anticipate may improve its performance.

Finally, the graph in Fig. 3 compares TPLEX for the SA dataset, in two configurations: with a combination of labelled and unlabelled documents as usual, and with only labelled documents. In both instances the algorithm was given the same seed and testing documents. In the first case the algorithm learned patterns using both the labeled and unlabeled documents. However, in the second case, only the labeled documents were used to generate the patterns. These data confirm that TPLEX is indeed able to improve performance from unlabelled data.

## 5 Discussion

We have described TPLEX, a semi-supervised algorithm for learning information extraction patterns. The key idea is to exploit the following recursive definition: good patterns are those that extract good fragments, and good fragments are those that are extracted by good patterns. This definition allows TPLEX to perform well with very little training data in domains where other approaches that assume fragment redundancy would fail.

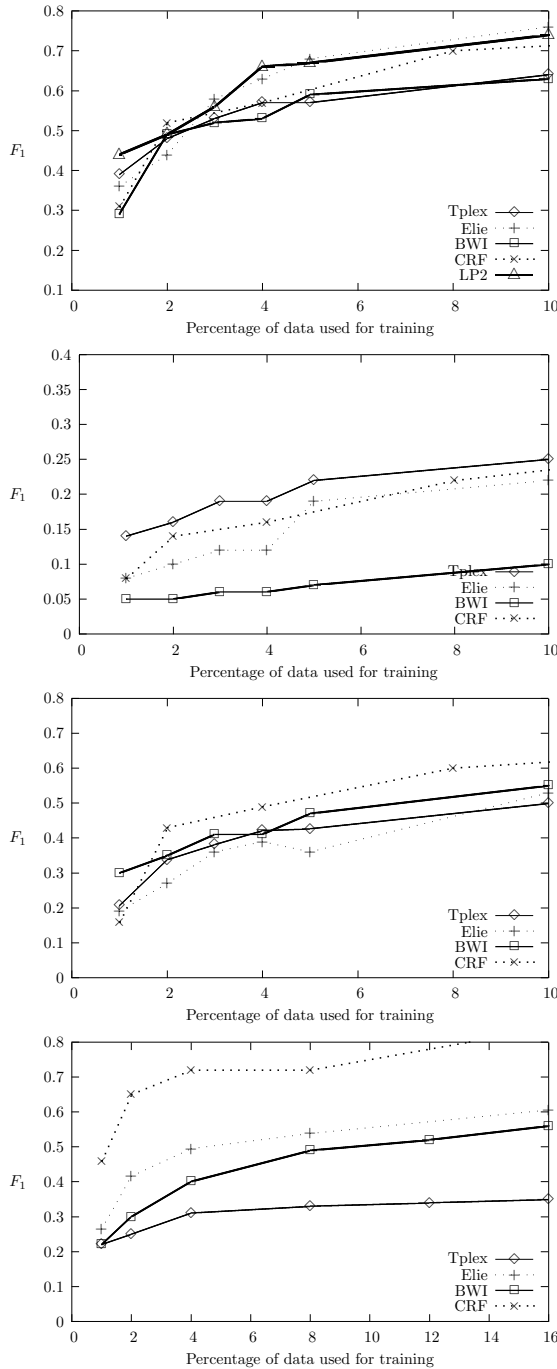


Figure 2: F1 averaged across all fields, for the Seminar (top), Acquisitions (second), Jobs (third) and MUC-NE (bottom) corpora.

**Conclusions.** From our experiments we have observed that our algorithm is particularly competitive in scenarios where very little labelled training data is available. We contend that this is a result of our algorithm’s ability to use the unlabelled test data to validate the patterns learned from the training data.

We have also observed that the number of fields that are being extracted in the given domain affects the performance of our algorithm. TPLEX extracts all fields simultaneously and uses the scores from each of the

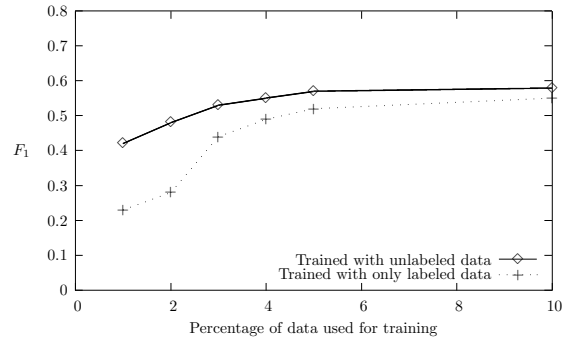


Figure 3: F1 averaged across all fields, for the Seminar dataset trained on only labeled data and trained on labeled and unlabeled data

patterns that extract a given position to determine the most likely field for that position. With more fields in the problem domain there is potentially more information on each of the candidate positions to constrain these decisions.

**Future work.** We are currently extending TPLEX in several directions. First, position filtering is currently performed as a distinct post-processing step. It would be more elegant (and perhaps more effective) to incorporate the filtering heuristics directly into the position scoring mechanism. Second, so far we have focused on a BWI-like pattern language, but we speculate that richer patterns permitting (for example) optional or re-ordered tokens may well deliver substantial increases in accuracy.

We are also exploring ideas for semi-supervised learning from the machine learning community. Specifically, probabilistic finite-state methods such as hidden Markov models and conditional random fields have been shown to be competitive with more traditional pattern-based approaches to information extraction (Fuchun and McCallum, 2004), and these methods can exploit the Expectation Maximization algorithm to learn from a mixture of labelled and unlabelled data (Lafferty et al., 2004). It remains to be seen whether this approach would be effective for information extraction.

Another possibility is to explore semi-supervised extensions to boosting (d’Alché Buc et al., 2002). Boosting is a highly effective ensemble learning technique, and BWI uses boosting to tune the weights of the learned patterns, so if we generalize boosting to handle unlabelled data, then the learned weights may well be more effective than those calculated by TPLEX.

**Acknowledgements.** This research was supported by grants SFI/01/F.1/C015 from Science Foundation Ireland, and N00014-03-1-0274 from the US Office of Naval Research.

## References

- E. Agichtein, L. Gravano, J. Pavel, V. Sokolova, and A. Voskoboynik. 2001. Snowball: A prototype system for extracting relations from large text collections. In *Proc. Int. Conf. Management of Data*.
- A. Blum and T. Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proc. 11th Annual Conference on Computational Learning Theory*, pages 92–100.
- S. Brin. 1998. Extracting patterns and relations from the World Wide Web. In *WebDB Workshop at the Int. Conf. Extending Database Technology*.
- M. E. Califf and R. Mooney. 1999. Relational learning of pattern-match rules for information extraction. In *Proc. American Nat. Conf. Artificial Intelligence*.
- C. Cardie. 1997. Empirical methods in information extraction. *AI Magazine*, 18(4).
- F. Ciravegna. 2001. Adaptive information extraction from text by rule induction and generalisation. In *Proc. Int. J. Conf. Artificial Intelligence*.
- F. Ciravegna. 2003. LP<sup>2</sup>: Rule induction for information extraction using linguistic constraints. Technical report, Department of Computer Science, University of Sheffield.
- M. Collins and Y. Singer. 1999. Unsupervised models for named entity classification. In *Proc. joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 100–110.
- F. d’Alché Buc, Y. Grandvalet, and C. Ambroise. 2002. Semi-supervised MarginBoost. In *Proc. Neural Information Processing Systems*.
- O. Etzioni, M. Cafarella, D. Downey, A. Popescu, T. Shaked, S. Soderland, D. Weld, and A. Yates. 2005. Unsupervised named-entity extraction from the Web: An experimental study. *Artificial Intelligence*. In press.
- A. Finn and N. Kushmerick. 2004. Multi-level boundary classification for information extraction. In *Proc. European Conf. Machine Learning*.
- D. Freitag and N. Kushmerick. 2000. Boosted wrapper induction. In *Proc. American Nat. Conf. Artificial Intelligence*.
- P. Fuchun and A. McCallum. 2004. Accurate information extraction from research papers using conditional random fields. In *Proc. Human Language Technology Conf.*
- N. Kushmerick and B. Thomas. 2003. Adaptive information extraction: Core technologies for information agents. *Lecture Notes in Computer Science*, 2586.
- J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. Int. Conf. Machine Learning*.
- J. Lafferty, X. Zhu, and Y. Liu. 2004. Kernel conditional random fields: Representation, clique selection, and semi-supervised learning. In *Proc. Int. Conf. Machine Learning*.
- A. McCallum. 2002. Mallet: A machine learning for language toolkit. <http://mallet.cs.umass.edu>.
- E. Riloff and R. Jones. 1999. Learning dictionaries for information extraction by multi-level bootstrapping. In *Proc. American Nat. Conf. Artificial Intelligence*.
- E. Riloff. 1993. Automatically constructing a dictionary for information extraction tasks. In *Proc. American Nat. Conf. Artificial Intelligence*.
- E. Riloff. 1996. Automatically generating extraction patterns from untagged text. In *Proc. American Nat. Conf. Artificial Intelligence*.
- S. Soderland. 1999. Learning information extraction rules for semi-structured and free text. *Machine Learning*, 34(1-3):233–272.
- M. Stevenson and M. Greenwood. 2005. Automatic learning of information extraction patterns. In *Proc. 19th Int. J. Conf. on Artificial Intelligence*.
- V. Vapnik. 1998. *Statistical learning theory*. Wiley.
- R. Yangarber, W. Lin, and R. Grishman. 2002. Unsupervised learning of generalised names. In *Proc. Int. Conf. Computational Linguistics*.
- D. Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Meeting of the Association for Computational Linguistics*, pages 189–196.